GROUP TESTING WITH GREEDY ALGORITHM

Venkata Sai Pavan Vineeth Mathapati

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2021

APPROVED:

Hua Sun, Major Professor
Ifana Mahbub, Committee Member
Kamesh Namuduri, Committee Member
Shengli Fu, Chair of the Department of Electrical Engineering
Hanchen Huang, Dean of the College of Engineering
Victor Prybutok, Dean of the Toulouse Graduate School

Mathapati, Venkata Sai Pavan Vineeth. *Group Testing with Greedy Algorithm*. Master of Science (Electrical Engineering), August 2021, 29 pp., 19 numbered references.

Group testing is all about identifying properties of a set of elements by testing them.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Dr. Hua Sun, my major advisor and mentor, for his unwavering support and guidance. As an experienced academician and mentor, his insightful guidance guided me through my academic career at UNT. I was able to push myself to reach goals I did not think I was capable of under his guidance. Hopefully, I would also make him proud.

Dr. Ifana Mahbub and Dr. Kamesh Namuduri have been invaluable in directing me through my study and providing useful perspectives.

Finally, I would like to express my gratitude to all of my friends who have helped me through this difficult time in my life. I would like to thank Sri Srujan Gollapudi, Varun Kumar Jagini in particular for their support and for being a part of my life.

TABLE OF CONTENTS

Page

CHAPTER 1

INTRODUCTION

1.1 Background and History

In statistics and combinatorial mathematics, group testing refers to any method that defines the assignment of finding components of a set which have certain properties into tests on groups of items, as opposed to on singular components. It came into existence when it was first studied by Robert Dorfman in 1943, it is a relatively new field of mathematics that is a functioning space of exploration today and has a lot of scope in a wide range of pragmatic applications.

An illustrative example of group testing includes a series of lights which are associated together where we precisely know one of the lights is broken. The goal is to track down the broken light making fewest number of tests (a test is when we connect some of the bulbs to a power supply). A straightforward approach is to test every bulb separately. Nonetheless, when there are an enormous number of bulbs, we can stay significantly more productive in the event by pooling the bulbs into groups. For instance, in one go, if we associate the half of the bulbs, immediately we can figure out the group that has the broken bulb, precluding half of the bulbs in only one test [1].

Plans for completing such group testing can be rudimentary or intricate and the tests required at each stage might be disparate. Procedures in which the tests for the following stage rely upon the aftereffects of the past stages are called adaptive procedures, while schemes designed so that all the tests are known previously are called non-adaptive group testing. Pool design comes into play in the latter type of procedure to determine the structure of tests.

Group testing has applications in statistics, biology, computer science medicine and engineering. Present day interest in these testing plans has been revived by the Human Genome Project.

In contrast to, many areas of Mathematics, the roots of group testing can be traced back to a single report written by a single person: Robert Dorfman. The inspiration emerged during the Second World War when the United States Public Health Service and the Selective help set out upon an enormous scope undertaking to get rid of all syphilitic men called up for enlistment.

Testing a person for syphilis includes drawing a blood test from them and afterward breaking down the sample to determine the presence or absence of syphilis. Be that as it may, at that point, playing out this test was costly, and testing each trooper independently would have been exceptionally cost hefty and wasteful [1].

Assume there are n soldiers, this strategy for testing paves way for n separate tests. If an enormous number of individuals are contaminated, this technique would be sensible. However, in the more likely case that only a small number of the men are contaminated, a substantially more effective testing plan can be accomplished. The plausibility of a more compelling hinges on the following property: we can pool the soldiers into groups, and in each group, we can combine blood samples together. We can then test the combined sample to check if at least one soldier in the group has syphilis. This is the focal thought behind group testing. If one or more of the soldiers in this group has syphilis, a test is squandered (more tests should be performed to discover which soldier(s) it was). Then again, assuming nobody in the pool has syphilis, numerous tests are saved, since we can takeout each soldier in that group with only one test [1].

1.2 Objective of the Research

The primary objective of the thesis rests in comparing combinatorial group testing with

probabilistic group testing. The process of each group testing technique is within an algorithm which employs logarithmic conversions and sets up limits depending on the number of patients.

Combinatorial group testing is more inclined towards adaptive group testing and is mostly dealt with number of samples collected and pooled manually. The goal in combinatorial group testing is to minimize the number of tests required in a worst-case scenario. Probabilistic group testing brings in conditions like entropy, maximizing the information gain to determine the pooling patterns that go into testing. In probabilistic models, the defected items are expected to follow some probability distribution and the aim is to minimize the tests to find out defective items [2].

Pooling the samples into groups is also a task as it determines the rapidness of testing. With a basic idea over the sample which holds considerable probability for defectiveness is taken with a sample that has the least probability of being defective and pooled into a group. Thus, we concentrate here on multiple factors that influence the group testing.

1.3 Organization of the Thesis

The rest of the thesis has been organized into several chapters for better understanding and easy readability.

- Chapter 2 reads categorization of group testing.
- Chapter 3 explains combinatorial group testing.
- Chapter 4 gives an idea about probabilistic group testing.
- Chapter 5 explains about group testing using greedy algorithm and entropy.
- Chapter 6 shows the algorithm implementation.
- Chapter 7 deals with conclusion and future work realizations.
- Chapter 8 cites all the references taken.

CHAPTER 2

CATEGORIZATION OF GROUP TESTING

2.1 Further Advancements

Before we start off with categorization, a proper knowledge on the inception of the concept and its considerable development across needs to be studied. As we first studied above, Robert Dorfman in 1942 put up the idea to implement to test soldiers for syphilis. Since then, group testing was updated by esteemed individuals over the years by introducing various algorithms. Dorfman's report – likewise with all the early work on group testing – zeroed in on the probabilistic issue and intended to utilize the clever thought of group testing to lessen the normal number of tests expected to get rid of all syphilitic men in each pool of soldiers. The strategy was basic: placed the soldiers into groups of a given size, and utilize individual testing (testing things in gatherings of size one) on the positive groups to discover which were contaminated. Dorfman organized the ideal group sizes for this system against the prevalence rate of defectiveness in the population [3].

After 1943, group testing remained largely untouched for several years. At that point in 1957, Sterrett delivered an enhancement for Dorfman's methodology [4]. This newer process begins by again performing individual testing on the positive gatherings yet stopping when a positive test is recognized. At that point, the excess samples in the group are tried together, since almost certainly, none of them are positive [1].

The main careful treatment of group testing was given by Sobel and Groll in their developed 1959 paper regarding the matter [5]. They described five new procedures – in addition to generalizations for when the prevalence rate is unknown – and for the optimal one, they provided an explicit formula for the expected number of tests it would use. The paper also made the connection between group testing and information theory for the first time, as well as discussing

several generalizations of the group-testing problem and providing some new applications of the theory [1]. They state:

> One chemical apparatus is available, and the devices are tested by putting x of them (where $1 < x < n$) in a bell jar and testing whether any of the gas used in constructing the devices has leaked out into the bell jar. It is assumed that the presence of gas in the bell jar indicates only that there is at least one leaker and that the amount of gas gives no indication of the number of leakers. [2]

Sobel and Groll likewise referenced other mechanical applications like testing condensers and resistors, the fundamental thought is very much shown by the very same Christmas tree lighting issue that we studied. A group of lights is electrically orchestrated in arrangement and tried by applying a voltage across the entire bunch or any subset thereof. On the off chance that the lights are on, entire tried subset of bulbs should all be acceptable; assuming the lights are off, in any event one bulb in the subset is deficient [2].

2.2 Types of Group Testing

There are two independent classifications for group-testing problems; every group-testing problem is either adaptive or non-adaptive, and either probabilistic or combinatorial. This can also be versatile and non-versatile because of their nature of testing.

In probabilistic models, the defective items are assumed to follow some likelihood and the aim is to limit the normal number of tests expected to distinguish the defectiveness of each item.

On the other hand, with combinatorial group testing, the goal is to limit the number of tests needed in a 'worst-case scenario' that is, create a minmax algorithm and no prior idea of the distribution of defectives is assumed [2].

The other arrangement, adaptivity, concerns what data can be utilized while picking which things to group into a test. In general, the decision of which things to test can rely upon the aftereffects of past tests, as in the above light bulb issue. A calculation that returns by playing out

a test, and afterward utilizing the outcome (and every single past outcome) to choose which next test to perform, is called adaptive [1] [2].

Conversely, in non-adaptive algorithms, all tests are chosen ahead of time. This thought can be summed up to multistage calculations, where tests are partitioned into stages, and each test in the following stage should be chosen ahead of time, with just the information on the consequences of tests in past stages. Albeit adaptive procedure offers considerably more opportunity in design, it is realized that adaptive group testing calculations don't develop more than a constant factor in the number of tests required to identify the set of defective items [6].

In addition to this, non-adaptive methods are often valuable n light of the fact that one can continue with progressive tests without first investigating the aftereffects of every past test, taking into consideration the effective distribution of the testing process.

2.3 Conceptual Understanding

There are numerous approaches to broaden the issue of group testing. Quite possibly the most significant is called noisy group testing and manages a major assumption of the original problem: that testing is error-free. A group-testing problem is called noisy when there is some opportunity that the result of a group test is fallacious (e.g., comes out positive when the test contained no defectives) [1].

Group testing can be extended by taking situation in which there are more than two potential results of a test. For instance, a test may have the results 0,1 and 2+ corresponding to there being no defectives, a single defective, or an obscure number of defectives bigger than one. More generally, it is feasible to consider the result set of a test to be 0,1, k+ for some $k \in N$ [2].

Another explanation is to consider mathematical limitations on which sets can be tried. The above light issue is an illustration of this sort of limitation: just bulbs that show up sequentially

can be tested. Also, the things might be orchestrated all around, or in general, a net, where the tests are available paths on the graph. Another sort of mathematical limitation would be on the most extreme number of things that can be tried in a group, or the group sizes may be even, etc. Likewise, it could be helpful to consider the limitation that any given item can just show up in a specific number of tests [2].

There are unlimited approaches to keep remixing the essential rudimentary formula of group testing. The following elaborations will give an idea of a portion of the more exotic variations. In the 'good–mediocre–bad' model, each item is one of 'good', 'mediocre' or 'bad', and the result of a test is the type of the 'worst' item in the group. In threshold group testing, the aftereffect of a test is positive if the quantity of defective items in the group is more prominent than some limit or proportions [7].

Group testing with restrictions is a variant with applications in molecular biology. Here, there is a second class of things called inhibitors, and the aftereffect of a test is positive on the off chance that it contains at least one defective and no restrictions [8].

Ultimately, these all derivations and methods or employing techniques to determine the results of group testing fall under 2 distinct types.

- Combinatorial group testing
- Probabilistic group testing

CHAPTER 3

COMBINATORIAL GROUP TESTING

3.1 Formalization of Combinatorial Group Testing

- The input vector $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots\ldots, \boldsymbol{x_n})$ is defined to be a binary vector of length $\boldsymbol{n}$ i.e., $\boldsymbol{x} \in \{\boldsymbol{0}, \boldsymbol{1}\}^{\boldsymbol{n}}$ with the j-th item being called defective if and only if $\boldsymbol{x_j} = \boldsymbol{1}$. Further, any non-defective item is called a 'good' item [1].

- $\boldsymbol{x}$ is intended to describe the (unknown) set of defective items. The key property of $\boldsymbol{x}$ is that it is an implicit input. There is no direct knowledge of what the entries of $\boldsymbol{x}$ are, other than that which can be inferred via some series of 'tests'. This leads on to the next definition [1].

- Let $\boldsymbol{x}$ be an input vector. A set, $\boldsymbol{S} \subset \{\boldsymbol{1}, \boldsymbol{2}, \cdots, \boldsymbol{n}\}$ is called a test. When testing is noiseless, the result of a test is positive when there exists $\boldsymbol{j} \in \boldsymbol{S}$ such that $\boldsymbol{x_j} = \boldsymbol{1}$, and the result is a negative otherwise. Therefore, the goal of group testing is to come up with a method for choosing a 'short' series of tests that allow $\boldsymbol{x}$ to be determined, either exactly or with a high degree of certainty [1].

- A group-testing algorithm is said to make an error if it incorrectly labels an item (that is, labels any defective item as non-defective or vice versa). This is not the same thing as the result of a group test being incorrect. An algorithm is called zero-error if the probability that it makes an error is zero [1].

- ***t(d,n)*** denotes the minimum number of tests required to always find ***d*** defectives among n items with zero probability of error by any group-testing algorithm. For the same quantity but with the restriction that the algorithm is non-adaptive, the notation $\bar{\boldsymbol{t}}\boldsymbol{(d,n)}$ is used [1].

Group testing was first studied in the combinatorial context by Li in 1962 [9], with the introduction of Li's s-stage algorithm [2]. Li proposed an extension of Dorfman's '2-stage algorithm' to an arbitrary number of stages that required no more than $\boldsymbol{t} = \frac{\boldsymbol{e}}{\boldsymbol{\log_2 e}}\, \boldsymbol{d \log_2 n}$ tests to be guaranteed to d or fewer defectives among n items. The thought was to eliminate every one of the items in negative tests and separate the leftover items into groups as was done with the initial pool. This was to be done s-1 times before performing individual testing [1] [2].

Combinatorial group testing in general was later concentrated even more completely by Katona in 1973 [10]. Katona was more inclined towards matrix representation for non-adaptive group testing and proposed a procedure for finding the defective in the non-adaptive 1-defective case in no more than $\boldsymbol{t = [(\log_2 n)]}$ tests which he also proved to be ideal [1] [2].

In general, discovering ideal calculations for adaptive combinatorial group testing is troublesome, and albeit the computational intricacy of group testing has not been resolved, it is suspected to be hard in some intricacy class [2]. As time passed by, in 1972, there occurred a pivotal quantum leap, with the introduction of the generalized binary-splitting algorithm [11]. The generalized binary-splitting algorithm works by performing a binary search on groups that test positive and is a simple algorithm that finds a single defective in no more than the information-lower-bound number of tests [1].

There also are cases, where more than 2 defectives can occur. Binary Splitting algorithm is still capable of producing near-optimal results in such cases. requiring at most $\boldsymbol{d - 1}$ tests above the information lower bound where $\boldsymbol{d}$ is the number of defectives.

3.2 Li's S-stage Algorithm:

Li was the first to study combinatorial group testing. He was worried about the circumstance where mechanical and logical tests are directed uniquely to figure out which of the factors are significant. Usually, only a relatively small number of critical variables exists among a large group of candidates. These critical variables are assumed to have effects too large to be masked by the experimental error, or the combined effect of the unimportant variables. Interpreting each variable as an item, each critical variable as a defective, and each experiment as a group test, a large effect from an experiment indicates the existence of a critical variable among the variables covered by the experiment. Li assumed that there are exactly d critical variables to start with and

set to minimize the worst-case number of tests. Since Li, CGT has been studied alongside with PGT for those classical applications in medical, industrial and statistical fields. Recently, CGT is also studied in complexity theory, graph theory, learning models, communication channels and fault tolerant computing. While it is very encouraging to see a wide interest in group testing, one unfortunate consequence is that the results obtained are fragmented and submerged in the jargons of the fields [2].

As we proceed with the algorithm, Li extended a 2-stage algorithm of Dorfman (for PGT) to s stages. At stage 1, the n items are arbitrarily divided into g_1 groups of k_1 items. Each of these groups is tested and items in pure groups are identified as good and removed. Items in contaminated groups are pooled together and arbitrarily redivided into g_2 group of k_2 items; thus, entering stage 2.

In general, at stage i, $2 \leq i \leq s$, items from the contaminated groups of stage $i-1$ are pooled and arbitrarily divided into g_i groups of k_i items, and a test is performed on each such group. k_s is set to be 1; thus, every item is identified at stage s. Let t_s denote the number of tests required by Li's S-stage algorithm [2].

Note that s = 1 corresponds to the individual testing algorithm, i.e., testing the items one by one. Thus $t_1 = n$. Next consider s = 2. For easier analysis, assume that n is divisible by k_1. Then

$$\boldsymbol{t_2 = g_1 + g_2 \leq \frac{n}{k_1} + dk_1}$$

Ignoring the constraint that k_1 is an integer, the upper bound is minimized by setting $\mathbf{k_1} = \sqrt{\frac{n}{d}}$. This gives $\boldsymbol{g_1 = \sqrt{nd}}$ and $\boldsymbol{t_2 = 2\sqrt{nd}}$.

Now consider the general s case.

$$\boldsymbol{t_s = \sum_{i=1}^{s} g_i \leq \frac{n}{k_1} + \frac{dk_1}{k_2} + \cdots \cdot \frac{+ dk_{s-2}}{k_{s-1}} + dk_{s-1}}.$$

Again, ignoring the integral constraints, then the upper bound is minimized by

$$k_i = \left(\frac{n}{d}\right)^{\frac{s-1}{s}},\ 1 \le i \le s-1.$$

This gives,

$$g_i \le d\left(\frac{n}{d}\right)^{\frac{1}{s}}$$

And

$$t_s \le s\,d\left(\frac{n}{d}\right)^{\frac{1}{s}}$$

Li gave numerical solutions for such s for given values of n/d.

To execute the algorithm, one needs to compute the optimal s and k_i for $i = 1, \ldots, s$. Each k_i can be computed in constant time. Approximating the optimal s by the ceiling or floor function of $\log n/d$, then Li's S-stage algorithm runs in $O(\log n/d)$ time.

We now show the surprising result that Li's s-stage algorithm can be implemented as a 3-bin algorithm. The three bins are labeled "queue," "good item" and "new queue." At the beginning of stage i, items which have been identified as good are in the good-item bin, and all other items are in the queue bin. Items in the queue bin are tested in groups of size k, (some possibly $k_i - 1$) as according to Li's S-stage algorithm.

Items in groups tested negative are thrown into the good-item bin, and items in groups tested positive are thrown into the new-queue bin. At the end of stage i, the queue bin is emptied and changes labels with the new-queue bin to start the next stage. Of course, at stage s, each group is of size one and the items thrown into the new-queue bin are all defectives [2].

3.3 Hwang's Generalized Binary Splitting Algorithm:

It is notable that one can recognize a defective from a contaminated group of n items in $[\log(n)]$ tests through *binary splitting*. Namely, segment the n items into two disjoint groups such that neither group has size exceeding $2^{[\log n]-1}$. Test one such group, the result shows either the

tested group or the other one is debased. Apply binary splitting on the new polluted group. A recursive argument shows that in $[\log(n)]$ tests a contaminated group of size 1 can be obtained, i.e., a defective is found. A special binary splitting method is the dividing strategy which segments the two groups as equitably as could really be expected [1] [2].

By applying binary splitting d times, one can distinguish the d defectives in the (d, n) issue in all items considered **d[log n]** tests. Hwang recommended an approach to facilitate the d applications of binary splitting to such an extent that the number of tests are diminished [2].

The thought is, generally, that there exists in normal an inadequate in each n/d items. Rather than getting a polluted group of size about half of the original group, which is the soul of binary splitting, one could hope to get a lot more modest polluted group and thus to identify a defective therein in fewer number of tests [1].

The generalized binary-splitting algorithm is an essentially optimal adaptive group-testing algorithm that finds d or fewer defectives among n items as follows:

1. If $n \leq 2d-2$, test the n items individually. If $n \geq 2d-2$, set $l = n-d+1$. Define $\alpha = [\log(\frac{l}{d})]$.

2. If $n > 2d-2$, test a group of size 2^{α}. If the outcome is negative, the 2^{α} items in the group are identified as good. Set $n := n-2^{\alpha}$ and go to step 1. If the outcome is positive, use binary splitting to identify one defective and an unspecified number, say x, of good items. Set $n := n-1-x$ and $d := d-1$. Go to Step 1.

Since it takes constant time to compute α , the generalized binary splitting algorithm can be solved in $O\,d(log\,n/d)$ time [1] [2].

The generalized binary-splitting algorithm requires no more than T tests where

$$T = \begin{cases} n & n \leq 2d-2 \\ (\alpha+2)d+p+1 & n \geq 2d-1 \end{cases} \qquad [2]$$

CHAPTER 4

NON-ADAPTIVE ALGORITHMS AND PROBABILISTIC GROUP TESTING

Non-adaptive group-testing algorithms will in general expect that the quantity of defectives, or at least a good upper bound on them, is known. This amount is signified $\boldsymbol{d}$ in this segment. On the off chance that no limits are known, there are non-adaptive algorithms with low inquiry intricacy that can help gauge $\boldsymbol{d}$ [1].

4.1 Combinatorial Orthogonal Matching Pursuit:

Combinatorial Orthogonal Matching Pursuit, or COMP, is a simple non-adaptive group-testing algorithm that frames the reason for the more convoluted algorithms that continue in this segment [1].

First, each entry of the testing matrix is chosen i.i.d. to be 1 with probability 1/d and 0 otherwise [1].

The decoding step proceeds column-wise (i.e., by item). If every test in which an item appears is positive, then the item is declared defective; otherwise, the item is assumed to be non-defective. Or equivalently, if an item appears in any test whose outcome is negative, the item is declared non-defective; otherwise, the item is assumed to be defective. An important property of this algorithm is that it never creates false negatives, though a false positive occurs when all locations with ones in the j^{th} column of M (corresponding to a non-defective item j) are "hidden" by the ones of other columns corresponding to defective items [1].

The COMP algorithm requires no more than $\boldsymbol{ed(1+\delta)ln(n)}$ tests to have an error probability less than or equal to $\boldsymbol{n^{-\delta}}$ [12]**.** This is within a constant factor of the lower bound for the average probability of error above [1].

4.2 Definite Defectives:

The definite defectives method (DD) is an augmentation of the COMP calculation that endeavors to eliminate any bogus positives. Execution ensures for DD have been appeared to stringently surpass those of COMP [13].

The decoding step utilizes a valuable property of the COMP calculation: that each item that COMP announces non-damaged is positively non-faulty (that is, there are no bogus negatives). It continues as follows.

1. First the COMP algorithm is run, and any non-defectives that it identifies are taken out. All leftover things are presently "possibly defective".
2. Next the algorithm takes a gander at all the positive tests. Assuming a thing shows up as the solitary "possible defective" in a test, it should be inadequate, so the calculation pronounces it to be flawed.
3. All other items are thought to be non-inadequate. The legitimization for this last advance comes from the suspicion that the quantity of defectives is a lot more modest than the absolute number of things [1].

Note that steps 1 and 2 never make a mistake, so the algorithm can only make a mistake if it declares a defective item to be non-defective. Thus, the DD algorithm can only create false negatives [1].

4.3 Sequential COMP:

SCOMP (Sequential COMP) is an algorithm that utilizes the way that DD commits no errors until the last step, where it is expected that the leftover items are non-defective. Let the arrangement of announced defectives be $\boldsymbol{K}$. A positive test is called explained by $\boldsymbol{K}$ if it contains at least one item in $\boldsymbol{K}$. The vital perception with SCOMP is that the arrangement of defectives found by DD may not clarify each sure test, and that each unexplained test should contain a secret inadequate [1].

The algorithm proceeds as follows.

1. Do stages 1 and 2 of the DD calculation to acquire ***K***, an underlying assessment for the arrangement of defectives.

2. On the off chance that ***K*** clarifies each certain test, end the calculation: ***K*** is the last gauge for the arrangement of defectives.

3. On the off chance that there are any unexplained tests, track down the **"possible defective"** that shows up in the biggest number of unexplained tests, and proclaim it to be damaged (that is, add it to the set ***K***). Go to step 2.

In simulations, SCOMP has been appeared to perform near ideally [1].

CHAPTER 5

ADAPTIVE GROUP TESTING USING ENTROPY AND MAXIMIZATION

We study the issue typically alluded to as group testing in the context of COVID19. Given n samples gathered from patients, how might we select and test combinations of tests to amplify data and limit the quantity of tests? Group testing is very much considered issue with a few engaging arrangements, however ongoing natural investigations force viable imperatives for COVID-19 that are contrary with conventional techniques. Besides, existing strategies utilize pointlessly prohibitive arrangements, which were conceived for settings with more memory and compute constraints rather than emphasizing the current issue. This results in poor utility [14].

Lacking powerful medicines or immunizations, the best method to save lives in a progressing pandemic is to moderate and control its spread. This should be possible by testing and secluding positive cases adequately early to forestall resulting diseases. Whenever done routinely and for an adequately enormous part of defenseless people, mass testing can possibly forestall many diseases a positive case would typically cause. However, several factors, such as limits on material and human resources, necessitate economical and efficient use of test resources [14].

Group testing aims to improve test quality by testing groups of samples simultaneously. We wish to leverage this framework to design practical and efficient COVID-19 tests with limited testing resources. Group testing can be adaptive or non-adaptive. In the former, tests can be decided one at a time, considering previous test results. In the latter, one can run tests in parallel, but also must select all tests before seeing any lab results [14].

A popular example of a semi-adaptive group test is to first split n samples into g groups of (roughly) equal size, pool the samples within the groups and perform g tests on the pooled samples.

All samples in negatively tested pools are marked as negative, and all samples in positively tested pools are subsequently tested individually [14].

5.1 Practical Constaints

Although group testing is an all-around contemplated issue, the new COVID-19 pandemic presents explicit requirements. As opposed to seroprevalence immunizer tests, PCR tests plan to distinguish dynamic cases, and just effectively do as such during part of the infection course. This results in a small prevalence (prior probability of population infection; we will assume a default value of 10-3), accepting we screen everybody as opposed to just suggestive people. Group testing has recently been validated for COVID-19 PCR tests. It is worked with by the way that PCR is an enhancement method that can identify little infection focuses. By and by, there are constraints on the quantity of tests l that can be set in a group, and limitations on the occasions a specific sample can be used. Besides, there are practical issues: adaptive testing is time consuming and hard to manage. Complex multiplex designs are prone to human error [16] [17].

5.2 Novel Formulation

We detail the issue dependent on the rule of data acquire given n individuals and m testing packs, the attributes of the test and earlier probabilities for every individual to be wiped out, we try to streamline the way the tests are utilized by consolidating a few examples. For effortlessness, tests are thought to be independent. However, we focus on implementable tests, unlike [18] which focuses on asymptotic results that are valid for large n [14].

5.3 Optimal Characterization

Despite the simplicity, it turns out that this greedy strategy has exponential running time and becomes infeasible for $n \geq 16$ [14].

5.4 Preliminaries

Notations are progressively introduced throughout but are gathered in the appendix, which also contains the proofs. Denote the number of patient samples by n. As previously mentioned, we consider the group testing task in the context of the COVID-19 pandemic. This choice of problem setting naturally introduces new mathematical constraints of a practical nature [14].

5.5 Impracticality of Adaptivity

Adaptive methods require several hours in between each lab result of the adaptive sequence. This inspires us to only consider either non-adaptive methods or semi-adaptive methods with no more than two phases of testing [14].

5.6 Low Concentration and Test Accuracy

Unnecessary blending of patient swabs may bring about restrictively low popular focus with unfortunate results for testing. A new report reports that one can securely blend a patient swab up to 10 times [16]; another relays that mixing up to 32 patient samples into the same probe yields a false negative rate below 10% [17] [14].

5.7 Notations and Reminders

Denote the number of tests to run by m. Tests are assumed to be imperfect, with a true positive rate (or sensitivity) tpr (equivalent terms include hit rate, detection rate and recall) and true negative rate (or specificity) tnr (equivalent terms include correct rejection rate and selectivity). As simple default values, we will use tpr = 99%, and tnr = 90% (This number is influenced by choice predisposition since it vigorously relies upon the phase of the infection; it is lower if an individual is tried past the point of no return our outcomes give direction with respect

to how to examine the examples that were gathered as opposed to the assortment timing and convention itself) [19] [17].

Patient sample i is infected with probability $p_i \in [0,1]$ and we assume statistical independence of infection of patient samples. Denoting by a '1' a positive result (infection), the unknown ground truth is a vector of size n made up of '0's and '1's. This vector describes who is infected and who is not. We call this the *secret*, denoted as $s \in [0,1]^n$. A design of a test $d \in [0,1]^n$ to run in the lab is a subset of patient samples to mix into the same sample, where $d_i = 1$ if patient sample i is mixed into design d and $d_i = 0$, otherwise. Note that the outcome of a perfect design d for a given secret s can simply be obtained as $1_{\langle d,s\rangle>0}$ where $\langle d, s\rangle \coloneqq \sum_{i:1}^{n} d_i s_i$. That is, a test result is positive if there is at least one patient i for which $d_i = 1$ (patient i is included in the sample) and $s_i = 1$ (patient i is infected) [14].

Recall that the secret s is unknown. However, since we assume that patient sample i is infected with probability p_i and that patient samples are independent, we have a prior probability distribution over the possible values of s. We hence represent the random value of s as a random variable (r.v) denoted by S, with probability distribution $p_S(s) \coloneqq Pr[S = s]$ over $[0,1]^n$. Let us now recall the definition of the entropy of our random variable [14].

$$H(S) = -\sum_{s\in\{0,1\}^n} p_S(s) \log_s p_S(s)$$

The entropy represents the amount of uncertainty that we have on its outcome, measured in bits. It is maximized when S follows a uniform distribution and minimized when S constantly outputs the same value. As we perform tests, we gain additional knowledge about S. For instance, if we group all samples into the same pool and have a negative result, then our posterior probability that all patients are healthy goes up, That is $p_S((0,\ldots,0))$ increases according to Bayes' rule of

probability theory. More generally, we may perform a sequence of tests of varying composition, updating our posterior after each test. Our goal will be to select designs of tests to minimize entropy, resulting in the least amount of uncertainty about the test outcome for all individuals [14].

5.8 Solving for Small Number of Patients

Given n people, test characteristics tpr & tnr and a set of prior probabilities of sample infection $(p_i)_{1\leq i\leq n}$, the best multiset D of m pool designs is the one maximizing the information gain. The tests are order insensitive, which gives a search space of cardinality $\left(\binom{2^{n+m}}{m}\right)$. Evaluating the information gain of every multiset separately take $O\ (2^{n+m})$ operations. Hence, brute-forcing this search space is prohibitive even for small values of n and m [14].

5.9 Motivating Greedy Information Strategy

Note that since tests are imperfect, for a given pool design $d \in [0,1]^n$ and a given secret $s \in [0,1]^n$ the Boolean outcome $T(s,d)$ of the test in the lab is not deterministic. If tests were perfect, we would have $T(s,d) = 1_{\langle d,s\rangle>0}$. To allow for imperfect tests, we model $T(s,d)$ as a r.v. whose distribution is described by $Pr[T(s,d) = \{1|\langle d,s\rangle > 0\}]$ = tpr and $Pr[T(s,d) = \{0|\langle d,s\rangle = 0\}]$ = tnr. Since the secret s is also unknown (and described by the r.v. *S*), the outcome $T(s,d)$ has now two sources of randomness: imperfection of tests and unknown secret. In practice, one will not run one test but multiple tests. We now suppose that m tests of pool designs are run and let their designs be represented as a multiset $D \in (\{0,1\}^n)^m$ [14].

This leads us to the following question: given an initial prior probability distribution p_S over the secret, how should we select pool designs to test in the lab? We want to select it such that once we have its outcome, we have as much information as possible about *S*, i.e., the entropy

(uncertainty) of *S* has been minimized. Since we cannot know in advance the outcome of the tests, we must minimize this quantity in expectation over the randomness coming from both the imperfect test and unknown secret. This requires the notion of conditional entropy [14].

5.10 Conditional Entropy

Given pool designs D, we consider two random variables S (secret) and $T := T(S, D)$ (test results). The conditional entropy of S given T is given by:

$$H(S|T) = -\sum_{s\in\{0,1\}^n t\in\{0,1\}^n} Pr[S = s, T = t] \log_s Pr[S = s, T = t]/\, Pr[T = t]$$

In this formula, the joint probability $Pr[S = s, T = t]$ has been computed with the conditional probability formula $Pr[S = s, T = t] = Pr[S = s]\ Pr[T = t|S = s]$, and the posterior distribution is computed using Bayesian updating, i.e.,

$$p_{S|T=t}(s) = 1\ Pr[S = s|T = t] = 1\, Pr[S = s, T = t]/\, Pr[T = t],$$

where $Pr[T = t = \Sigma_s Pr[S = s, T = t]$. It represents the amount of information (measured in bits) needed to describe the outcome of *S*, given that the result of *T* is known. The mutual information between *S* and *T* can equivalently be defined as $I(S, T) := H(S) - H(S|T)$. It quantifies the amount of information obtained about *S* by observing *T* [14].

A well-motivated criterion for test selection: Since *H(S)* does not depend on *d*, selecting the pool design d minimizing the conditional entropy of *S* given the outcome of *D* is equivalent to selecting the one maximizing the mutual information between *S* and *T (S, D)*. We now have a clear criterion for selecting *D* [14]:

$D^* \in arg\ max\ I(S, T(S, D))$ over *D*

This criterion selects the pool designs *D* whose outcome will maximize our information about *S*.

CHAPTER 6

RESULTS

```
import numpy as np


# the following input data is taken initially, it can be changed accordingly
n = 10   // # initializing the total number of patient samples (n),
m = 5   // # total number of tests to run (m), true positive rate (tpr),
tpr = 99 // # true negative rate (tnr) and initial list of patient sample of
tnr = 90 // # probabilities that if they have the infection or not (p)


# ps is the vector of size n, representing the person who is infected by '1' and vice versa
# therefore s is clearly is a subset of {0, 1}^n
ps = [0.1, 0.4, 0.7, 0.2, 0.9, 0.0, 0.3, 0.5, 0.6, 0.8]
Pr_S = np.array([i for i in ps if i != 0]) // # Pr_S here represents Pr[S=s]
Pr_T = np.copy(Pr_S) // # Pr_T here represents Pr[T=t]


# Pr_T_g_S here represents Pr[S=s|T=t]
# Pr_T_g_S = np.empty([n])
Pr_T_g_S = np.array([0.1, 0.4, 0.7, 0.2, 0.9, 0.3, 0.5, 0.6, 0.8])


# Pr_S_g_T here represents Pr[T=t|S=s]
# values filled for testing purposes, they must be calculated beforehand
# Pr_S_g_T = np.empty([n])
Pr_S_g_T = np.array([0.1, 0.4, 0.7, 0.2, 0.9, 0.3, 0.5, 0.6, 0.8])
```

```
# setting the zero filtered array size
n = len(Pr_S)


# entropy of the random variable S
def entropy(S):


# H(S): equation - (1)
 return np.sum((-1)*S*np.log2(S))


# conditional entropy of S given the T (test results)
def conditional_entropy(Pr_S, Pr_S_g_T, Pr_T, n):


# H(S|T): equation - (2)
    ce = 0
    for s in range(n):
        for t in range(n):
            ce += (Pr_S[s]*Pr_S_g_T[t])*np.log2(Pr_S[s]*(Pr_S_g_T[t]/Pr_T[t]))
    return -ce

# Maximum Likelihood Outcome
def ML():


# ML(t): equation - (5)
    return np.max((Pr_S*Pr_T_g_S)/np.sum(Pr_S*Pr_T_g_S))
```

```
# Algorithm 1: (Greedy-Adaptive)
def Greedy_Adaptive(n = n, m = m, tpr = tpr, tnr = tnr, pi = ps):
    k = m
    ps = pi

    while k > 0:



# calculating entropy of S
        ent_S = entropy(Pr_S)
        print(f'H(S): {ent_S}')



# calculating conditional entropy (H(S|T))
        ent_st = conditional_entropy(Pr_S, Pr_S_g_T, Pr_T, n)
        print(f'H(S|T): {ent_st}')

# computing I(S, T) = H(S) - H(S|T)
        I = np.absolute(ent_S - ent_st)
        print(f'I(S, T): {I}\n')

# then the arg max is selected from I
# and the result is observed for T(S, d*) after
# selecting a new design based on I and new ps is updated as
# ps = [new data]
# ps is taken as pi just for testing
        ps = pi
         k -= 1



Greedy_Adaptive()
```

Results

- Test 1: I(S, T): 7.08633
- Test 2: I(S, T): 6.36487
- Test 3: I(S, T): 5.64345
- Test 4: I(S, T): 4.92646
- Test 5: I(S, T): 3.55087

CHAPTER 7

CONCLUSION AND FUTURE WORK

We have introduced a structure for group testing considering points of interest of the current COVID19 pandemic. It applies techniques for likelihood and data hypothesis to build and unravel multiplex codes crossing the significant scope of group sizes. Use of tools like python to compute posterior probabilities and tally them to achieve information gain is precise. This way, we can minimize the channel of false positives and achieve the true number of positives in less time. Usage of resources and exacerbating them will only make things worse during the time of global pandemic. Also, unprecedented times like these are so seldom in the world, so usage of group testing for not so deadly diseases is also recommended. Maximization technique that is applied over information gain will eliminate the human errors that can be made with combinatorial group testing [14].

Future work includes we accept that the test multiplexing issue is an ideal chance for our world to make a commitment towards tending to the current worldwide emergency. By solidly establishing this issue in learning and surmising strategies, we give prolific ground to additional turn of events. As more data about test attributes opens up, we could consider conditions of tpr, tnr on pool size. The system could be adjusted to various target capacities, or connected to choose hypothesis utilizing reasonable danger functionals, e.g., considering the downstream danger of misdiagnosing a person with specific attributes (comorbidities, likelihood of spreading the sickness, and so forth).

When used with multiple access channels, each client can tune in and send on the channel, however if more than one client communicates simultaneously, the signals impact, and are decreased to indiscernible clamor. Multiaccess channels are significant for different true

applications, eminently remote PC organizations and telephone organizations.

In the setting of group testing, this issue is typically handled by separating time into 'epochs' in the accompanying way. A client is called 'active' if they have a message toward the beginning of an epoch. (In the event that a message is produced during an epoch, the client just gets active toward the beginning of the following one.) An epoch closes when each active client has effectively sent their message. The issue is then to track down every one of the active clients in each epoch and timetable a period for them to communicate [1].

REFERENCES

[1] https://en.wikipedia.org/wiki/Group_testing

[2] Ding-Zhu, Du; Hwang, Frank K. (1993). Combinatorial group testing and its applications. Singapore: World Scientific. ISBN 978-9810212933.

[3] Dorfman, Robert (December 1943), "The Detection of Defective Members of Large Populations", The Annals of Mathematical Statistics, 14 (4): 436–440,

[4] Sterrett, Andrew (December 1957). "On the detection of defective members of large populations". The Annals of Mathematical Statistics. 28 (4): 1033–1036. doi:10.1214/aoms/1177706807

[5] Sobel, Milton; Groll, Phyllis A. (September 1959). "Group testing to eliminate efficiently all defectives in a binomial sample". Bell System Technical Journal. 38 (5): 1179–1252. doi:10.1002/j.1538-7305.1959.tb03914.x

[6] Atia, George Kamal; Saligrama, Venkatesh (March 2012). "Boolean compressed sensing and noisy group testing"

[7] Chen, Hong-Bin; Fu, Hung-Lin (April 2009). "Nonadaptive algorithms for threshold group testing". Discrete Applied Mathematics. 157 (7): 1581–1585.

[8] De Bonis, Annalisa (20 July 2007). "New combinatorial structures with applications to efficient group testing with inhibitors". Journal of Combinatorial Optimization. 15 (1): 77–94. doi:10.1007/s10878-007-9085-1

[9] Li, Chou Hsiung (June 1962). "A sequential method for screening experimental variables". Journal of the American Statistical Association. 57 (298): 455–477. doi:10.1080/01621459.1962.10480672.

[10] Katona, Gyula O.H. (1973). "A survey of combinatorial theory". Combinatorial Search Problems. North-Holland, Amsterdam: 285–308.

[11] Hwang, Frank K. (September 1972). "A method for detecting all defective members in a population by group testing". Journal of the American Statistical Association. 67 (339): 605–608. doi:10.2307/2284447. JSTOR 2284447.

[12] Chun Lam Chan; Pak Hou Che; Jaggi, Sidharth; Saligrama, Venkatesh (1 September 2011), "2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)", 49th Annual Allerton Conference on Communication, Control, and Computing, pp. 1832–1839, arXiv:1107.4540, doi:10.1109/Allerton.2011.6120391, ISBN 978-1-4577-1817-5

[13] Aldridge, Matthew; Baldassini, Leonardo; Johnson, Oliver (June 2014). "Group Testing Algorithms: Bounds and Simulations". IEEE Transactions on Information Theory. 60 (6): 3671–3687. arXiv:1306.6438. doi:10.1109/TIT.2014.2314472.

[14] Abraham, L., Becigneul, G., Coleman, B., Scholkopf, B., Shrivastava, A., & Smola, A. (2020). Bloom Origami Assays: Practical Group Testing. *arXiv preprint arXiv:2008.02641*.

[15] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. Journal of Artificial Intelligence Research, 42:427–486, 2011.

[16] Michael Schmidt, Sebastian Hoehl, Annemarie Berger, Heinz Zeichhardt, Kai Hourfar, Sandra Ciesek, and Erhard Seifried. FACT - Frankfurt adjusted COVID-19 testing - a novel method enables high-throughput SARS-CoV-2 screening without loss of sensitivity

[17] Idan Yelin, Noga Aharony, Einat Shaer-Tamar, Amir Argoetti, Esther Messer, Dina Berenbaum, Einat Shafran, Areen Kuzli, Nagam Gandali, Tamar Hashimshony, Yael Mandel-Gutfreund, Michael Halberthal, Yuval Geffen, Moran Szwarcwort-Cohen, and Roy Kishony. Evaluation of COVID-19 RT-qPCR test in multi-sample pools. Clinical Infectious Diseases, 2020.

[18] Arya Mazumdar. Nonadaptive group testing with random set of defectives. IEEE Transactions on Information Theory, 62(12):7522–7531, 2016.

[19] Nikhil S Padhye. Reconstructed diagnostic sensitivity and specificity of the rt-pcr test for covid-19. medRxiv, 2020